PSN implies SN

Emmanuel Polonovski

PSN Implies SN

Emmanuel Polonovski

PPS, CNRS - Université Paris 7
Emmanuel.Polonovski@pps.jussieu.fr

Abstract. In the framework of explicit subsitutions there is two termination properties: preservation of strong normalization (PSN), and strong normalization (SN). Since there are not easily proved, only one of them is usually established (and sometimes none). We propose here a connection between them which helps to get SN when one already has PSN. For this purpose, we formalize a general proof technique of SN which consists in expanding substitutions into "pure" λ-terms and to inherit SN of the whole calculus by SN of the "pure" calculus and by PSN. We apply it successfully to a large set of calculi with explicit substitutions, allowing us to establish SN, or, at least, to trace back the failure of SN to that of PSN.

1 Introduction

Calculi with explicit substitutions were introduced [1] as a bridge between λ-calculus [7] and concrete implementations of functional programming languages. Those calculi intend to refine the evaluation process by proposing reduction rules to deal with the substitution mechanism – a *meta*-operation in the traditional λ-calculus. It appears that, with those new rules, it was much harder (and sometimes impossible) to get termination properties. The two main termination properties of calculi with explicit substitutions are:

- **Preservation of strong normalization** (PSN), which says that if a pure term (*i.e.* without explicit substitutions) is strongly normalizing (*i.e.* cannot be infinitely reduced) in the pure calculus (*i.e.* the calculus without explicit substitutions), then this term is also strongly normalizing with respect to the calculus with explicit substitutions.
- **Strong normalization** (SN), which says that, with respect to a typing system, every typed term is strongly normalizing in the calculus with explicit substitutions, *i.e.* every terms in the subset of typed terms cannot be infinitely reduced.

These two properties are not redundant, and Fig. 1 shows the differences between them. PSN says that the horizontally and diagonally hatched rectangle is included in the diagonally hatched rectangle. SN says that the vertically hatched rectangle is included in the diagonally hatched rectangle. Even if they work on a different set of terms, there is a common part: the vertically and horizontally hatched rectangle, wich represent the typed pure terms.

SN and PSN are both termination properties, although their proofs are not always clearly related: sometimes SN is shown independently of PSN (directly, by simulation, *etc.*, see for example [12,11]), sometimes SN proofs uses PSN (see for example [4]). We present here a general proof technique of SN via PSN, initially suggested by H. Herbelin, which uses that common part of typed pure terms.

In section 2, we formalize the technique and in section 3 we summarize the results we achieved by applying it to a set of calculi. This set has been choosen for the variety of

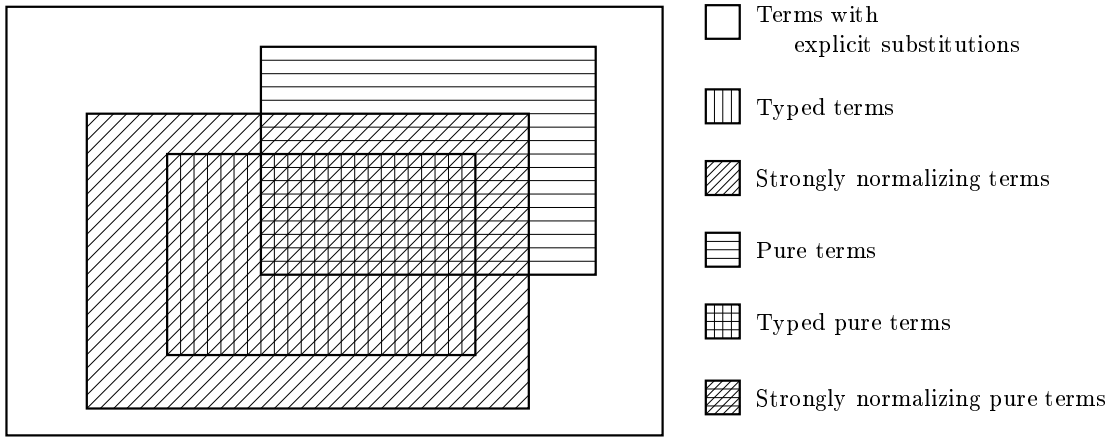

Fig. 1. Normalization properties of terms with and without explicit substitutions

their definitions: with or without De Bruijn indices, unary or multiple substitutions, with or without composition of substitutions, and even a symmetric non-deterministic calculus. In the last section, we briefly talk about perspectives in this framework.

2 Proof Technique

The idea of this technique is the following. Let t be a typed term with explicit substitutions for which we want to show termination. With the help of its typing judgment, we build a typed pure term t' which can be reduced to t. For that purpose, we expand the substitutions of t into redexes. We call this expansion *Ateb* (the opposite of *Beta* which is usually the name of the rule which creates explicit substitutions). Then, with SN of the pure calculus and PSN, we can export the strong normalization of t' (in the pure calculus) to t (in the calculus with explicit substitutions).

In practice, this sketch will only apply in some cases, and some others will require some adjustment to this technique. For our technique to work, we need that the *Ateb* expansion satisfies some properties. The first one is always easily checked.

Property 1 (Preservation of typability). If t is typable, with respect to a typing system T, in the calculus with explicit substitution, then $Ateb(t)$ is typable, with respect to a typing system T' (possibly $T' = T$) in the pure calculus.

Only some calculi can exhibit an *Ateb* function which satisfies the second one.

Property 2 (Initialization). $Ateb(t)$ reduces to t in zero or more steps in the calculus with explicit substitutions.

If we can get it, then we use the direct proof to be presented in section 2.1. Otherwise, we need to use the simulation proof to be presented in section 2.2. In the sequel, $\mathcal{SN}$ will be the set of strongly normalizing pure terms and $\mathcal{SN}_x$ will be the set of strongly normalizing terms of the calculus with explicit substitutions.

2.1 Direct proof

We can immediately establish the theorem.

Theorem 1. *For all typing systems T and T' such that, in the pure calculus, all typable terms with respect to T are strongly normalizing, if there exists a function Ateb from explicit substitution terms to pure terms satisfying properties 1 and 2 then PSN implies SN (with respect to T').*

Proof. For every typed term t of the calculus with explicit substitution, $Ateb(t)$ is a pure typed term (by property 1). By the strong normalization hypothesis of the typed pure calculus, we have $Ateb(t) \in \mathcal{SN}$. By hypothesis of PSN we obtain that $Ateb(t)$ is in $\mathcal{SN}_x$. By property 2, we get $Ateb(t) \to^* t$, which gives us directly $t \in \mathcal{SN}_x$.

2.2 Simulation proof

We must relax some constraints on $Ateb$. We will try to find an expansion of t to t' such that t' reduces to a term u and there exists a relation $\mathcal{R}$ with $u\mathcal{R}t$. The choosen relation must, in addition, enable a simulation of the reductions of t by the reduction of u. If it is possible, we can infer strong normalization of t from strong normalization of u.

To proceed with the simulation, we first split the reduction rules of the calculus with explicit substitutions into two disjoints sets. The set R_1 contains rules which are trivially terminating, and R_2 contains the others. Secondly, we build a relation $\mathcal{R}$ which satisfies the following properties.

Property 3 (Initialisation). For every typed term t, there exists a term $u\mathcal{R}t$ such that $Ateb(t)$ reduces in 0 or more steps to u in the calculus with explicit substitutions.

Property 4 (Simulation *). For every term t, if $t \to_{R_1} t'$ then, for every $u\mathcal{R}t$, there exists u' such that $u \to^* u'$ and $u'\mathcal{R}t'$.

Property 5 (Simulation $^+$). For every term t, if $t \to_{R_2} t'$ then, for every $u\mathcal{R}t$, there exists u' such that $u \to^+ u'$ and $u'\mathcal{R}t'$.

We display those properties as diagrams :

Initialisation	Simulation *	Simulation $^+$
t	$t \to_{R_1} t'$	$t \to_{R_2} t'$
$\swarrow \mathcal{R}$	$\mathcal{R} \quad \mathcal{R}$	$\mathcal{R} \quad \mathcal{R}$
$Ateb(t) \to^* u$	$u \to^* u'$	$u \to^+ u'$

With this material, we can establish the theorem.

Theorem 2. *For all typing systems T and T' such that, in the pure calculus, all typable terms with respect to T are strongly normalizing, if there exists a function Ateb from explicit substitution terms to pure terms and a relation $\mathcal{R}$ on explicit substitutions terms satisfying properties 1, 3, 4 and 5 then PSN implies SN (with respect to T').*

Proof. We prove it by contradiction. Let t be a typed term with explicit substitutions which can be infinitly reduced. By property 3 there exists a term u such that $Ateb(t) \rightarrow^* u$, and $Ateb(t)$ is a pure typed term (by property 1). By the strong normalization hypothesis of the typed pure calculus, we have $Ateb(t) \in \mathcal{SN}$. By hypothesis of PSN we obtain that $Ateb(t)$ is in $\mathcal{SN}_x$ and it follows that $u \in \mathcal{SN}_x$.

By property 3, we also have $u\mathcal{R}t$, and, with properties 4 and 5, we can build an infinite reduction from u, contradicting the strong normalization of u.

3 Results

3.1 λx-calculus

The λx-calculus [6,5] is probably the simplest calculus with explicit substitutions. It only makes the subtitution explicit. Since this calculus provides no rules to deal with substitutions composition, it preserves strong normalization. It is for this calculus that the technique has been originately used by Herbelin. Therefore, we can without surprises apply the direct proof to get strong normalization.

3.2 λυ-calculus

The $\lambda\upsilon$-calculus [16,3] is the De Bruijn counterpart of λx. As λx, it has no composition rules, and therefore satisfies PSN. For this calculus, we must use the simulation proof to deal with indices modification operators. We succeed to use it and it is, as far as we know, the first proof of SN for a simply typed version of $\lambda\upsilon$ (see [19]).

3.3 λ_{ws}-calculus

The λ_{ws}-calculus [13,9,10] introduces an explicit weakening operator, which allows to preserve strong normalization even with composition rules. It has already been shown to be SN [12]. We fail to apply the technique, due to the explicit weakening operator combined with the rigidity of the typing environment one usually has in calculi with De Bruijn indices.

3.4 λ_{wsn}-calculus

In [12] a named version of λ_{ws} was proposed. In current work, we developed a new version of this calculus : λ_{wsn}. We already have a SN proof for this calculus, almost similar to the original one, and this technique can be applied, using the direct proof. We cannot conclude to SN by this way, since PSN has not yet been shown (see [19]).

3.5 λσ-calculus

The well known $\lambda\sigma$-calculus [1] does not have either PSN nor SN, as shown in [17]. However, we can successfully apply our technique, using the simulation proof. It does not gives us SN, but it reduces the SN problem to that of PSN. If someone proposes a strategy which preseves strong normalization, our work will give immediately a SN proof.

3.6 $\lambda\sigma_n$-calculus

Introduced in the same work [1], the named version of $\lambda\sigma$ suffers the same problem concerning PSN. We can also apply the simulation proof to it, and conclude similarly.

3.7 $\overline{\lambda}\mu\tilde{\mu}\mathbf{x}$-calculus

The $\overline{\lambda}\mu\tilde{\mu}$-calculus [8,14] is a symmetric version of the $\lambda\mu$-calculus [18]. As for symmetric λ-calculus [2], the symmetry raises difficulties in normalization proofs. We can build an explicit substitutions version "à la" $\lambda\mathbf{x}$: $\overline{\lambda}\mu\tilde{\mu}\mathbf{x}$. In [20], we apply successfully the technique, by direct proof, to show its strong normalization.

4 Perspectives

It seems that this technique can be used for many calculi with explicit substitutions. Its application on named calculi is easy, in general, and leads to a simple direct proof. For some others, as for calculi with De Bruijn indices, we must use the simulation proof, which tend to be not so easy. Further work includes its application to the λ_{ws}-calculus and to the λlxr-calculus [15].

References

1. Abadi, M., Cardelli, L., Curien, P.-L., Lévy, J.-J.: Explicit Substitutions. Journal of Functional Programming (1991).
2. Barbanera, F., Berardi, S.: A symmetric lambda-calculus for classical program extraction. Proceedings of TACS'94 (1994), Springer-Verlag LNCS **789**, 495–515.
3. Benaissa, Z.-E.-A., Briaud, D., Lescanne, P., Rouyer-Degli, J.: λv, a calculus of explicit substitutions which preserves strong normalisation. Journal of Functional Programming (1996).
4. Bloo, R.: Preservation of Termination for Explicit Substitutions. PhD thesis, Eindhoven University (1997).
5. Bloo, R., Geuvers, H.: Explicit Substitution: on the Edge of Strong Normalisation. Theoretical Computer Science (TCS 1999), **211**, 375–395.
6. Bloo, R., Rose, K.: *Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection.* In Computing Science in the Netherlands, pages 62-72. Netherlands Computer Science Research Foundation, 1995.
7. Church, A.: The Calculi of Lambda Conversion. Princeton Univ. Press (1941).
8. Curien, P.-L., Herbelin, H.: The duality of computation. Proceedings of ICFP'00 (2000), ACM Press, 233–243.
9. David, R., Guillaume, B.: The λ_l-calculus. In D. Kesner, editor, Proceedings of the 2nd Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs, pages 2-13, July 1999.
10. David, R., Guillaume, B.: A λ-calculus with explicit weakening and explicit substitution. Mathematical Structures in Computer Science, 11, 2001.
11. David, R., Guillaume, B.: Strong Normalisation of the Typed λ_{ws}-calculus. In Proceedings of the 17th International Worshop Computer Science Logic (CSL 2003), volume 2803 of Lecture Notes in Computer Science, pages 155-168. Springer, Vienna, 2003.
12. Di Cosmo, R., Kesner, D., Polonovski, E.: Proof nets and explicit substitutions. In J. Tiuryn, editor, Foundations of Software Science and Computation Structures (FOSSACS 2000), volume 1784 of Lecture Notes in Computer Science, pages 63-81. Springer-Verlag, Mar. 2000.
13. Guillaume, B.: Un calcul de substitution avec étiquettes. PhD thesis, Université de Savoie (1999).

14. Herbelin, H.: Explicit substitutions and reducibility. Journal of Logic and Computation (2001), **11**, 429–449.
15. Kesner, D., Lengrand, S.: Broadening the horizon of the explicit substitution paradigm via a logical model. Submitted paper (2004).
16. Lescanne, P.: From lambda-sigma to lambda-upsilon: a journey through calculi of explicit substitutions. In 21st ACM Symposium on Principles of Programming Languages (POPL'94), 16-19 Janvier 1994, Portland, Oregon, pp 60-69.
17. Mellies, P.-A.: Typed λ-calculi with explicit substitutions may not terminate. Proceedings of TLCA'95 (1995), Springer LNCS, **902**, 328–334.
18. Parigot, M.: $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. Proceedings of LICS'93 (1993), Computer Society Press, 39–46.
19. Polonovski, E.: Substitutions explicites, logique et normalisation. PhD thesis, Université Paris 7, (2004). In preparation.
20. Polonovski, E.: Strong normalization of $\overline{\lambda}\mu\tilde{\mu}$-calculus with explicit substitutions. In Proceedings of Foundations of Software Science and Computation Structures (FOSSACS 2004), LNCS 2987, Mar. 2004.